Beyond Classical Limits: Quantum Vulnerabilities in Public Key Encryption

Pau Minguet Martí

*Computer Science*

*California Polytechnic State University*

*San Luis Obispo, CA*

[pminguet@calpoly.edu](mailto:pminguet@calpoly.edu)

***Abstract – This paper explores quantum computing and its implications for prime factorization and internet security, beginning with an overview of public-key encryption and outlining the challenges of prime factorization on classical computers. I will then introduce Shor's algorithm, which leverages quantum computing to rapidly factor large numbers, and explain it in detail, including relevant mathematical concepts. I will discuss the current state of quantum computing research, examining the potential threats to existing cryptographic protocols. Finally, I will explore alternative methods for secure online communications beyond public-key encryption based on the described quantum computing advancements.***

I. Introduction

Since its mainstream adaptation, the internet has relied on cryptography to secure the communication of its users. Before that, symmetric keys were necessary, and they could only be shared through a secure medium, which required both parties to meet in the real world in order to safely communicate through the web later. This method clearly does not support the internet as we know it, and communication was extremely limited until Ron Rivest, Adi Shamir, and Leonard Adleman created the first public key encryption algorithm. This transformed the world, and it is to this day one of the foundations that allow the world to function how it does. This new algorithm allowed people to have public keys that anyone could use to encrypt messages for them and a private key that allowed them to decrypt them, without ever having to share a key. The privacy of these keys relies on the difficulty of factoring a number into its two prime roots, which

becomes exponentially time-consuming as the number gets larger. Here is a short example I coded in Python, where a graph is generated to show the time it took my laptop to find the prime factors of numbers n of increasing size, from primes of 2 bits of length to 30 bits of length.

In this logarithmic graph, we clearly see that the larger the number becomes, the time it takes to find its prime factors grows exponentially. Additionally, the numbers used at this moment are around 600 digits long, which would be around 2000 bits, which would take the algorithm used to create the graph hundreds of millions of years to crack. Even the most powerful conventional supercomputers available would take thousands or even millions of years to break these encryptions, making it worthless to even attempt to.

III. Quantum Computing

Quantum Computing is an incredibly advanced and experimental field that attempts to harness the principles of quantum mechanics to perform computations. These principles, such as superposition or entanglement, allow us to perform operations very differently than transistor-based machines.

In quantum computing, the basic unit of information is the quantum bit or qubit, which can exist in a superposition of states (e.g., both 0 and 1 at the same time) unlike classical computing bits that can only represent 0 or 1. The ability of qubits to represent multiple states simultaneously, along with the phenomenon of quantum entanglement (where the state of one qubit is correlated with the state of another), allows quantum computers to perform certain computations exponentially faster than classical computers.

Even though these computers have the potential to make some extraordinarily hard problems trivial to solve, large-scale, fault-tolerant quantum computers remain a significant technological challenge.

IV. Shor’s Algorithm

Shor’s Algorithm, developed in 1994 by Peter Shor, is a groundbreaking quantum computing algorithm designed to solve the integer factorization problem and the discrete logarithm problem, both of which are crucial for modern encryption methods like RSA. It uses several mathematical principles including Euclid’s algorithm and the Fourier Transform to exploit the quantum properties of qubits to optimize an implementation that is far from optimal for conventional computers.

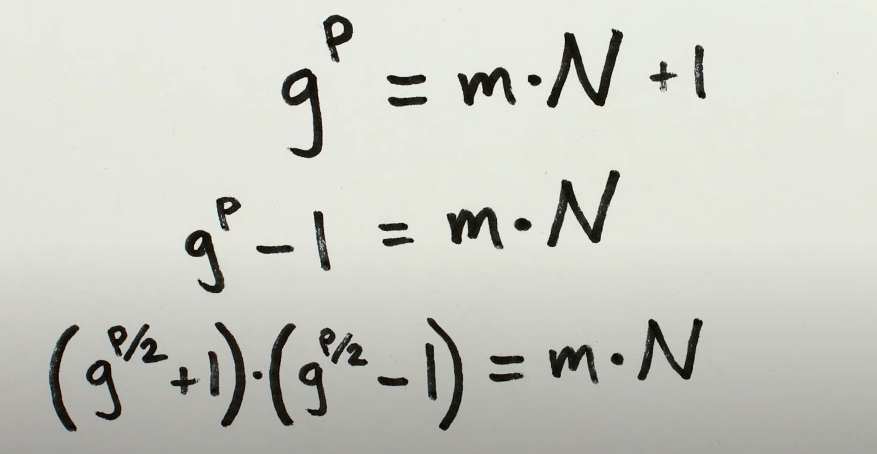
Since the interests behind breaking encryption are obviously strong, there exists a plethora of algorithms that attempt to decrease the time needed to accomplish it. But they are very complex and are only marginally more efficient than just guessing numbers from 1-**n** and checking whether it’s a factor of **n**. There are, however, some algorithms that use some mathematical principles to turn a **g** random (bad) guess into a much more promising one. On conventional computers, this turns out to be even slower than the old-school brute force approach. On quantum computers, however, this appears to be the right path.

The mathematical principle that attempts to turn a randomly generated number into a “good” guess, that is, one that has much greater odds of being a factor of **n**, is the following. For any pair of whole numbers that don’t share a factor, if you multiply one of them by itself enough times, you’ll eventually arrive at some whole number multiple of the other number plus 1. That is, if **g** and **n** are integers that don’t share factors, then eventually

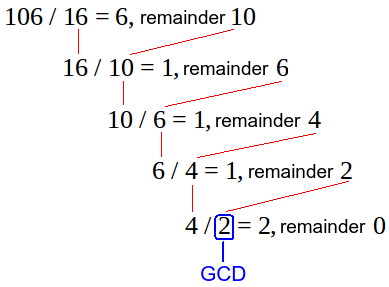
[**g**^**p** = **m**\***n** + 1], for some power **p** and some multiple **m**.

For example, g=7 and m=15. While seven squared isn’t one more than a multiple of 15, and neither is seven cubed, seven to the fourth is (7^4 = 160 \* 15 + 1). This same phenomenon occurs for any pair of numbers that don't share factors, although the power **p** will become a very large integer with the magnitudes that encryption deals with.

Once we have **p**, we can very easily modify that equation to show that **g**^(**p**/2) +1 and **g**^(**p**/2) -1 will be much better guesses as factors of **n** (As shown in the image below). In this algorithm, we will need an even integer for **p**, and if we were to get an odd number, the improved guess wouldn’t be an integer, so we would need to make another uninformed guess and recompute **p**. Once these improved guesses are made, they will likely not be factors of **n**, but they have much higher odds of sharing factors in common with **n**. Given this and other issues, these new improved guesses will lead to factoring **n** about 3/8ths of the time, which is a success rate of around 37.5%. This is a success rate of over 99% in under 10 guesses.



To check for common factors throughout the algorithm, we would use Euclid’s algorithm, which determines the Greatest Common Divisor (GCD) of our guesses and **n** in logarithmic time complexity. This simple algorithm just takes the larger number and divides it by the smaller number. Until the remainder is 0, we will keep dividing the divisor by the remainder. The GCD will be the last non-zero remainder. This image shows an example of the algorithm on 106 and 16.



Now, as explained earlier, finding **p** is a very computationally expensive task for conventional computers, but not for quantum computers. This is because qubits can represent a variety of states all at once, and only when they are measured is that we see one or zero. This superposition of states can be represented by a graph of probabilities. In an example of a single qubit, we could measure its state many times, and get an estimate of how often it collapses to state |0⟩ and |1⟩. For example, the state of a qubit that is 50% in the state |0⟩ and 50% in the state |1⟩ can be expressed as:

|ψ⟩ = (1/√2)|0⟩ + (1/√2)|1⟩

For a qubit that is 33.3% in the state |0⟩ and 66.6% in the state |1⟩, the quantum state can be expressed as:

|ψ⟩ = (1/√3)|0⟩ + (√2/√3)|1⟩

(Note that the sum of the squares of the coefficients adds up to 1)

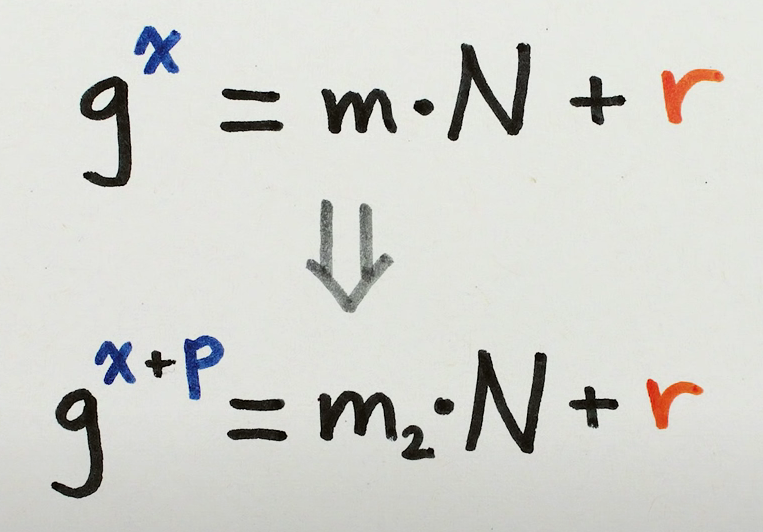
Using and measuring multiple qubits at once allows us to express larger numbers, in the same way that conventional bits do, except that in this case we see a probabilistic graph, with variable output when measuring the same system. For example, measuring 8 bits with state |ψ⟩ = (1/√2)|0⟩ + (1/√2)|1⟩ would result in a random number from 0 to 255. Quantum computers are the only currently known way to generate truly random numbers, as quantum mechanics still has many unknowns.

What this means, therefore, is that if we input into a function performed on qubits, that the input will be a probabilistic graph where, when we measure the state, we get a different singular input each time. When the function is computed on the array of qubits, it is performed on the probabilistic graph, that is, on all of the inputs at once, making quantum computers extraordinarily faster than conventional ones on trial-and-error problems.

For Shor’s algorithm, the first step is to calculate **g** to multiple powers of itself. This can be done by inputting a superposition of all integers we want to try out as potential exponents of **g** (1,2,3…) into a function that will raise **g** to those powers, and return a superposition of all the answers. Then we would input that into a function that returns the superposition of the difference between those and the closest multiple of **n**. That would leave us with a superposition of the remainders, along with their respective **p** values.

Given what was explained above, just measuring the resulting qubits will only return us a pair of a **p** value, and the remainder between **g**^**p** mod **n\*m**, which is extremely unlikely to be 1, which is what we’re looking for, essentially. To portray how unlikely it is that we measure the correct answer, take a value **n** to be a 600-digit integer, like we use in our current encryptions. That means that measuring the state will give us a correct **p** value, one that has 1 as the remainder, one in 10^600 times, which is no different than randomly guessing on a conventional computer.

The true potential of quantum computers, however, lies in the usage of destructive interference as we understand in the realm of waves, to make all of the wrong answers in our probabilistic graph output cancel each other, except the correct answer. To accomplish this, Peter Shor used this mathematical observation (image) to show that for any remainder **r** and exponent **x**, **g**^**x** and **g**^**x**+**p** have the same remainder (each with its own value of **m**, of course), which appear in a repeating pattern. This repeating property isn't something that could be figured out from taking our guess to just one power, it's a structural relationship between different powers, and we can take advantage of it since quantum computations can be performed on superpositions of many different powers.

****

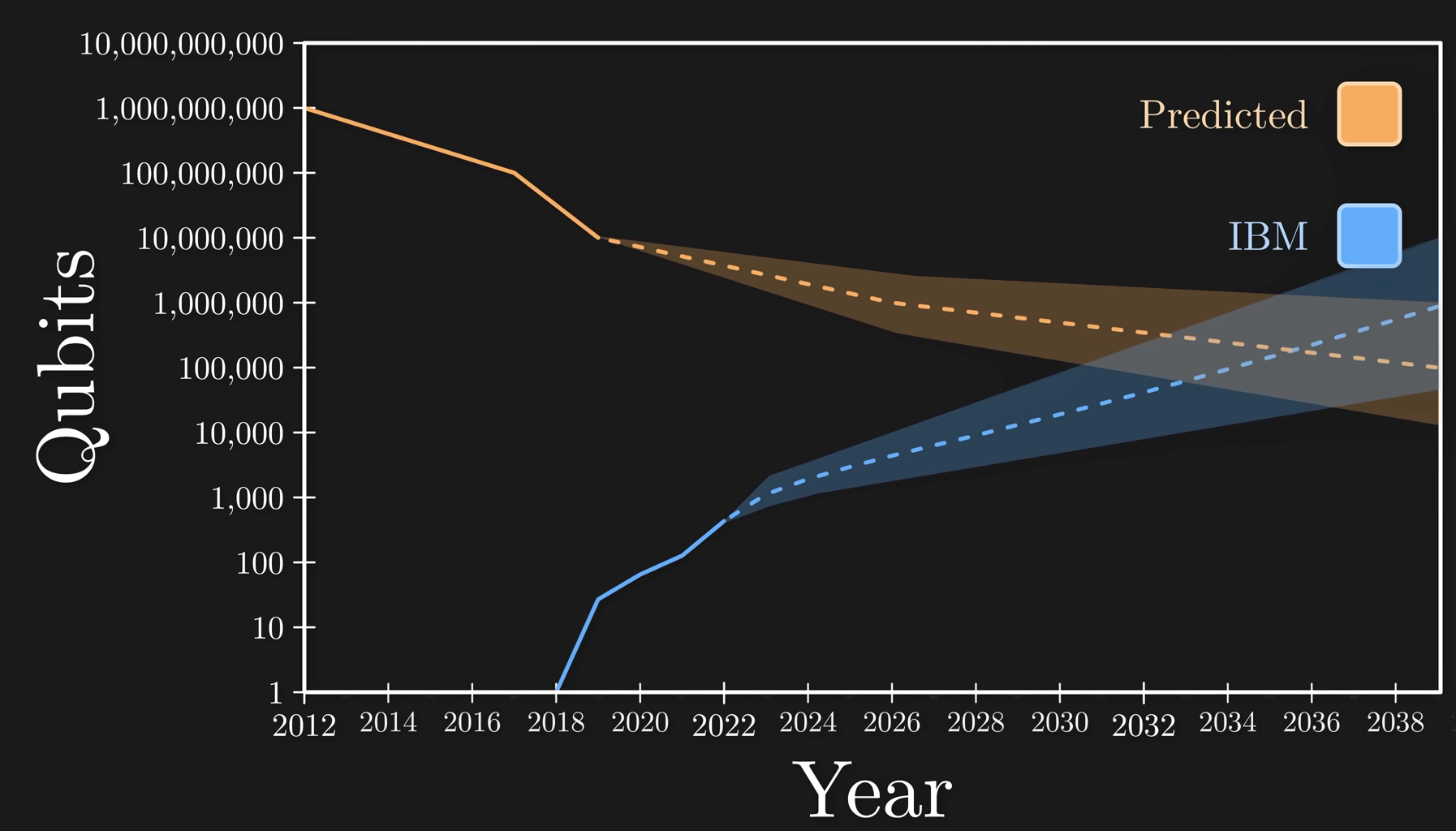
Specifically, if we take the superposition of all possible powers and only measure the remainder, then we'll randomly get one of the possible remainders, as explained above, as the output. For example, 3. The specific number is not important, but what is important is that we must be left with a superposition of purely the powers that could have resulted in a remainder of 3. This is another special properties of quantum computation.If you put in a superposition and get an answer that could have come from more than one element of the superposition, then you'll be left with a superposition of just those elements. In our case, because of the repeating property, those powers are all numbers that are **p** apart from each other (period). This means that the output probabilistic graph (wave), will behave like a very complicated sine function, one where the frequency is 1/**p**.

Once we have obtained this superposition of powers **p** which result in an equal remainder of a multiple of **n**, we perform a Fourier Transform. This mathematical tool is based on the idea that any periodic function can be represented as the sum of sinusoidal waves with different frequencies, amplitudes, and phases. It is widely used to decompose and remove noise from audio, analyzing and enhancing images, measuring the frequency components of light or other radiation, etc. This tool, however, can be used to take our superposition and output the one single frequency, 1/**p**, that separates all of the elements with equal remainder.

Once **p** is known, and having checked for an odd power, in which case it would necessitate to start over, we will have two very strong candidates to share factors with **n**: (**g^p**/2)+1 and (**g^p**/2)-1. Using Euclid’s Algorithm, we can rapidly check whether they share common denominators with **n**, and if they do, we will have broken the encryption.

V. Timelines, Scales, and Alternative Encryption Methods

Even though this simplified explanation of the foundations of Shor’s algorithm does prove that quantum computers could easily break our current encryption methods in minutes, the amount of qubits required to perform this algorithm on the magnitudes currently used is orders of magnitude larger than the current larges quantum computer in the world. Additionally, my synthesis of the functioning of qubits assumed no outside-world interferences, and in the current state of the field, this is still being worked on. As the investments toward this ground-breaking new technology grow, it is estimated that we will reach the number of qubits necessary to break current encryption within ten to fifteen years. The graph below shows the processing power of IBM’s most powerful quantum computer and the predicted amount of qubits needed for RSA key decryption.



Fortunately, the National Institute of Standards and Technology had a competition in 2016 in which they requested cryptographers for a quantum-resistant encryption method, and four were selected to be the future standard. How these encryption methods work is outside the scope of this paper, although the general idea of one of the most popular is the following. Keys will be vectors in an N-dimensional space, and their decryption would rely on finding coefficients for these vectors that most closely approximate to a point in this space, the options of which increase exponentially as we increase N. Currently, there are no algorithmic solutions that harness the unique powers of quantum computers to break this kind of encryption, but mathematicians still have a decade to come up with one.

VI. Conclusion

The development of quantum computing and Shor's algorithm poses a significant threat to current encryption methods like RSA that rely on the difficulty of factoring large numbers. While large-scale quantum computers capable of running Shor's algorithm are still years away, the implications for internet security and privacy are huge. Shor's algorithm harnesses quantum mechanics to exponentially speed up factoring large numbers, undermining the math behind RSA. The four encryption methods selected by NIST in 2016 are believed to be quantum-resistant for now, but may eventually be cracked as the field of quantum computing advances.

Governments, companies, and individuals all have an interest to stay ahead of this by continually developing new cryptographic techniques to withstand quantum computing's power. Collaboration between math, computer science, and physics researchers will be key. Embracing quantum's potential is vital for keeping online communications secure in the post-quantum world. While quantum computing challenges existing protocols, it's also an opportunity to innovate and explore the frontiers of cryptography. By actively researching quantum-resistant solutions, we can ensure secure online communication thrives in the quantum era.